VAX/VMS Access Control List Editor Reference Manual

Order Number: AA–Z414B–TE

April 1986

This document describes the VAX/VMS Access Control List Editor.

Revision/Update Information:	This revised document supersedes the *Access Control List Editor Reference Manual* (Order No. AA–Z414A–TE) Version 4.0
Software Version:	VAX/VMS Version 4.4

digital equipment corporation
maynard, massachusetts

April 1986

The postpaid READER'S COMMENTS form on the last page of this document requests the user's critical evaluation to assist in preparing future documentation.

The following are trademarks of Digital Equipment Corporation:

DEC	DIBOL	UNIBUS
DEC/CMS	EduSystem	VAX
DEC/MMS	IAS	VAXcluster
DECnet	MASSBUS	VMS
DECsystem–10	PDP	VT
DECSYSTEM–20	PDT	
DECUS	RSTS	
DECwriter	RSX	digital

ZK–3031

HOW TO ORDER ADDITIONAL DOCUMENTATION
DIRECT MAIL ORDERS

USA & PUERTO RICO*

Digital Equipment Corporation
P.O. Box CS2008
Nashua, New Hampshire
03061

CANADA

Digital Equipment
of Canada Ltd.
100 Herzberg Road
Kanata, Ontario K2K 2A6
Attn: Direct Order Desk

INTERNATIONAL

Digital Equipment Corporation
PSG Business Manager
c/o Digital's local subsidiary
or approved distributor

In Continental USA and Puerto Rico call 800-258-1710.

In New Hampshire, Alaska, and Hawaii call 603-884-6660.

In Canada call 800-267-6215.

*Any prepaid order from Puerto Rico must be placed with the local Digital subsidiary (809-754-7575).

Internal orders should be placed through the Software Distribution Center (SDC), Digital Equipment Corporation, Westminster, Massachusetts 01473.

This document was prepared using an in-house documentation production system. All page composition and make-up was performed by TEX, the typesetting system developed by Donald E. Knuth at Stanford University. TEX is a registered trademark of the American Mathematical Society.

ACL EDITOR Contents

FIGURES

Preface

Intended Audience

This manual is intended for all system users.

Structure of This Document

This document is composed of four major sections.

The Format Section is an overview of the Access Control List Editor and is intended as a quick-reference guide. The format summary describes the DCL command that invokes the ACL editor; the usage summary describes how to invoke and exit from the ACL editor, how to direct output, and restrictions you should be aware of.

The Description Section explains how to use the ACL editor and how to customize the VAXTPU ACL editor section file.

The Qualifier Section describes each DCL command qualifier. Qualifiers appear in alphabetical order.

The Examples Section contains examples of common operations that you perform with the ACL editor.

Associated Documents

To use the ACL editor, you should also be familiar with the following manuals:

- *VAX/VMS DCL Concepts Manual*
- *Guide to VAX/VMS System Security*

If you are a system programmer and plan on modifying the VAXTPU ACL section file from which the ACL editor was built, you should be familiar with the *VAX Text Processing Utility Reference Manual.* It describes the VAXTPU programming language and the concepts involved in modifying and processing a VAXTPU section file.

Conventions Used in This Document

The following conventions are observed in this manual:

Convention	Meaning
RET	A symbol with a one- to three-character abbreviation indicates that you press a key on the terminal, for example, RET.
CTRL/x	The phrase CTRL/x indicates that you must press the key labeled CTRL while you simultaneously press another key, for example, CTRL/C, CTRL/Y, CTRL/O. In examples, this control key sequence is shown as ^x, for example, ^C, ^Y, ^O, because that is how the system echoes control key sequences.
$ SHOW TIME 05-JUN-1982 11:55:22	Command examples show all output lines or prompting chracters that the system prints or displays in the black letters. All user-entered commands are shown in red letters.
$ TYPE MYFILE.DAT . . .	Vertical series of periods, or ellipsis, mean either that not all the data that the system would display in response to the particular command is shown or that not all the data a user would enter is shown.
file-spec,...	Horizontal ellipsis indicates that additional parameters, values, or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file specification or in the syntax of a substring specification in an assignment statement.)
quotation marks apostrophes	The term quotation marks is used to refer to double quotation marks ("). The term apostrophe (') is used to refer to a single quotation mark.

New and Changed Features

For VAX/VMS Version 4.4, the ACL editor has been rewritten using the VAX Text Processing Utility (VAXTPU). The following enhancements and changes are included in this version of the ACL editor:

- Support for a callable interface to the ACL editor. The callable ACL editor is documented in the *VAX/VMS Utility Routines Reference Manual.*
- Discontinued use of the initialization file SYS$LIBRARY:ACLEDIT.INI. Instead, make changes to the ACL editor VAXTPU section file SYS$LIBRARY:ACLEDIT.TPU.
- Support for dynamic ACE syntax checking.
- Discontinued support of non-ANSI terminals. Single character input is no longer necessary, improving the performance of the ACL editor.
- Improvements to screen management have been added. You can now shift text in the display window right or left by pressing GOLD → or GOLD ← . **(ACEs will no longer span multiple lines.)** You can also move the cursor back and forth through the ACL a screen at a time with the MOVE SCREEN key; the PREV SCREEN and NEXT SCREEN keys on VT200 series terminals provide the identical function.
- The RETURN key now performs the ACE entering function. (The ENTER keypad command also performs this function.)
- Support of a PASTE buffer for moving sections of the ACL. This feature is available only on VT200 series terminals and is, by default, normally disabled; Section 3.2.1 explains how to enable the PASTE buffer.

ACL EDITOR

Invokes the VAX/VMS Access Control List (ACL) Editor to create or modify an access control list for a specified object.

FORMAT

EDIT/ACL *object-spec*

Command Qualifiers	Defaults
/[NO]JOURNAL[=file-spec]	*See text.*
/MODE=option	*/MODE=PROMPT*
/OBJECT=type	*/OBJECT=FILE*
/[NO]RECOVER[=file-spec]	*/NORECOVER*

Command Parameter

object-spec

Specifies the object whose access control list is to be created or edited using the ACL editor. If the access control list does not exist, it is created. The object specified may be a file, directory, device, global section, or logical name table.

If the object is a file, the ACL editor does not provide a default file type. If you omit the file type, it is presumed to be null. The specified file must be a disk file on a Files–11 On-Disk Structure Level 2 formatted volume. If the object is a directory, specify a file specification with the file type of DIR. If the object type is anything other than a file or a directory, you must specify the /OBJECT=type qualifer.

No wildcard characters are allowed in the object specification.

usage summary

Invoking

You invoke the ACL editor with the DCL command EDIT/ACL.

The /ACL qualifier is required.

Exiting

You exit from the editing session normally, by pressing CTRL/Z. If the session ends abnormally and journaling has been in effect, you can recover the ACL with the /RECOVER qualifier. You can quit the editing session without saving the edits by pressing GOLD CTRL/Z.

Directing Output

By default, the editing session occurs at the SYS$OUTPUT device. As the result of a successful editing session, an access control list is created for an object. (The resultant ACL is kept as part of the file header.)

Privileges/Restrictions

A user can only invoke the ACL editor to create or modify the ACL for an object that the user either owns, has control access to, or can gain access to by a privilege such as BYPASS, GRPPRV, READALL, or SYSPRV.

A file or directory must currently exist and be on a Files–11 On-Disk Structure Level 2 disk.

ACL EDITOR

Description

DESCRIPTION

The ACL editor is a screen-oriented editor used to create and maintain access control lists (ACLs). Through the use of ACLs, you can define what types of access, if any, should be granted to the potential users of a system object.

By carefully defining the individual access control list entries (ACEs) that make up an ACL, user access to a particular object can be more closely controlled than through the use of default UIC-based protection.

The description of the ACL editor is divided into six parts:

- Section 1 provides an overview of Access Controls Lists (ACLs).
- Section 2 describes how to invoke the ACL editor.
- Section 3 displays the keypad commands available for use during an ACL editing session.
- Section 4 explains how to recover from an editing session that was interrupted abnormally.
- Section 5 describes the different types of ACEs available in an access control list.
- Section 6 describes how to customize the ACL editor.

1 An Overview of ACLs

An access control list consists of access control list entries (ACEs) that grant or deny access to a particular system object. ACLs may be placed on the following types of objects:

- Devices
- Files (including directory files)
- Group global sections
- Logical name tables
- System global sections

Because ACLs define access more precisely than the standard default protection or UIC-based protection, ACLs offer users the chance to refine the system's response when a user seeks access to an object. You can provide an ACL on any object to permit as much or as little access as is desirable in each case. Typically, ACLs are used when you want to provide access to a system object for some, but not all users on a system. ACLs can also dictate that security alarms be set off when unauthorized access to an object succeeds or fails.

Whenever the system receives a request for access to an object that has an ACL, the system searches each entry in the ACL from the first to the last for the first match it can find and then stops searching. If another match occurs further down in the ACL, it will not have any effect. Thus, if you intend to afford one or more users greater access through a particular ACE, you should position that ACE at or near the top of the ACL. More important ACEs identifying specific users should appear in the ACL before ACEs identifying broader classes of users.

The use of ACLs is optional. Although the use of ACLs can enhance the security of system objects in an installation through a more detailed definition of who is allowed what kind of access, user time must be spent in creating and maintaining the ACLs, and processor time is required to perform the functions that ACLs mandate.

Each ACL consists of one or more ACEs. There is no limit to the number of ACEs that an ACL can contain or to the number of characters in an ACE; however, very long ACLs increase the amount of time necessary to gain access to an object.

The three types of ACEs available in an ACL are *identifier, default protection,* and *security alarm* ACEs. Refer to Section 5 for a description of these ACEs.

2 Invoking the ACL Editor

The owner of a system object can define an ACL for that object with the ACL editor, or do extensive editing to an existing ACL. The ACL editor is invoked with the DCL command EDIT/ACL followed by the name of the object whose ACL you want to create or modify. For example, the following command invokes the ACL editor to create an ACL for the file INVENTORY.DAT:

```
$ EDIT/ACL INVENTORY.DAT
```

If the object whose ACL you want to create or modify is not a file, you must specify the type of object with the /OBJECT=type qualifier. For example, the following command invokes the ACL editor to create an ACL for the disk DOCD$:

```
$ EDIT/ACL/OBJECT=DEVICE DOCD$
```

You can also invoke the ACL editor with any of the following DCL commands:

- SET FILE/ACL/EDIT
- SET DIRECTORY/ACL/EDIT
- SET DEVICE/ACL/EDIT
- SET ACL/EDIT

The behavior of the ACL editor is the same regardless of which command invoked it. In addition to invoking the ACL editor, these commands can be used to manipulate entire ACLs or individual ACEs without invoking the ACL editor. For more information, see the *VAX/VMS DCL Concepts Manual.*

After the editor is invoked, the ACL appears on the screen if the object already has an ACL; otherwise, you will be creating the ACL by entering an access control list entry (ACE).

The ACL editor may be invoked from within a program written in any VAX language that generates calls using the VAX/VMS Calling Standard. Refer to the *VAX/VMS Utility Routines Reference Manual* for more information on using the callable interface to the ACL editor.

3 Keypad Editing

The ACL editor provides several features that simplify its use, including:

- Automatic text insertion (prompt mode)
- Dynamic ACE syntax checking
- Keypad editing that includes online help

By default, the ACL editor prompts for each ACE and provides values in the various fields within an ACE whenever possible. The /MODE qualifier controls the choice of mode. To disable prompting, specify /MODE=NOPROMPT.

The FIELD, ITEM, and ENTER commands on the keypad enable you to take full advantage of prompt mode in the ACL editor.

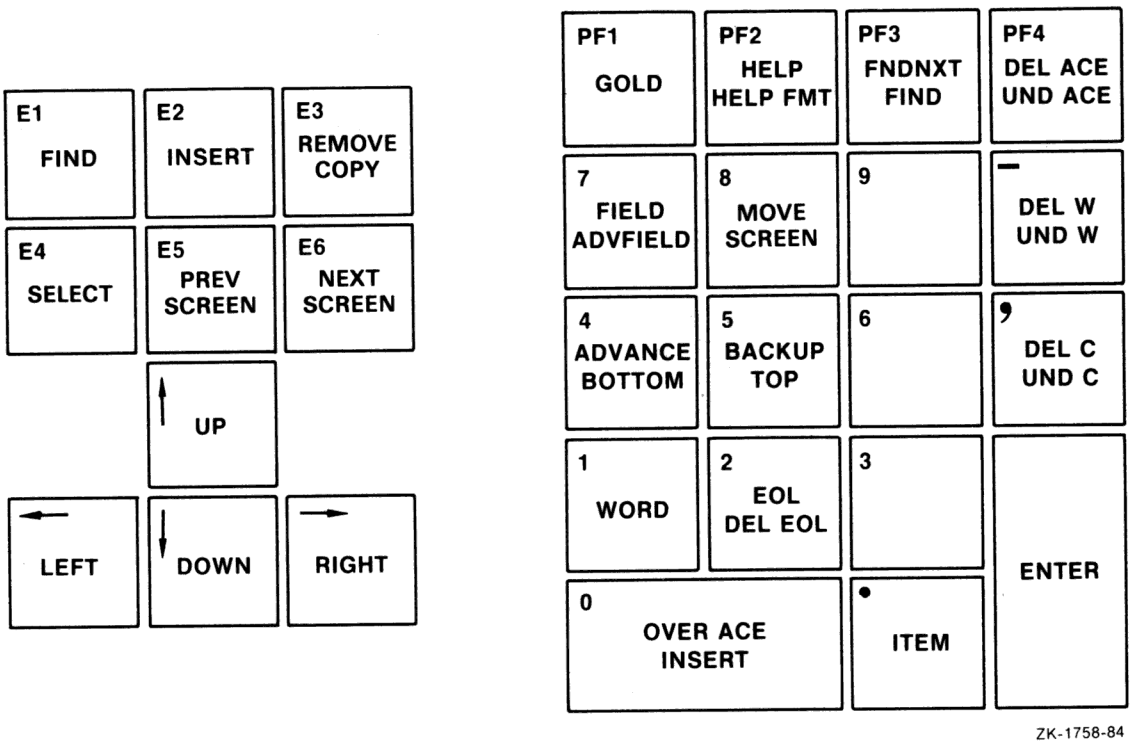
- FIELD—Completes the current ACE field and moves the cursor to the next ACE field or subfield, inserting text as needed. If the ACL editor is not in prompt mode, the ACL editor advances to the next field in the current existing ACE.
- ITEM—Selects the next item for the current ACE field. If the ACL editor is not in prompt mode, this key is ignored.
- ENTER—Indicates that the current ACE is complete and ready for parsing. This key terminates the insertion. You can press it while the cursor is located at any position within the ACE. (Performs the same function as the RETURN key.)

The online help facility of the ACL editor is accessed by using the HELP and HELP FMT commands on the keypad. The HELP command provides help on the editing keypad; the HELP FMT command provides help on ACE formats.

The following section contains a diagram of the default keypad functions for VT100 and VT200 series terminals. Also included in the section is the functional description of each of the editing commands available with the ACL editor. You can supplement or change these key definitions by modifying and recompiling the ACL editor section file SYS$LIBRARY:ACLEDIT.TPU. Refer to Section 6 for more information if you intend to modify the default ACL section file.

3.1 ACL Editing Commands

Figure ACL–1 VT200 Keypad

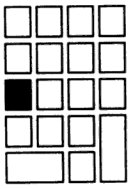


ZK-1758-84

Figure ACL–1 depicts the default ACL editor keypad functions for VT200 series terminals. The numeric keypad on VT100 series terminals is identical to that of the VT200 terminal shown above; VT100 terminals, however, do not have the supplemental editing keypad (keys E1 through E6).

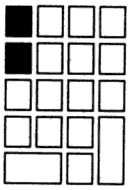
On the VT100 and VT200 series terminals, you can use the following keypad commands during an ACL editing session:

ADVANCE



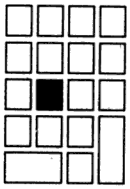
Sets the current direction forward for the FIND, FNDNXT, MOVE SCREEN, OVER ACE, and WORD keys. ADVANCE means that the editor's movement is toward the end of the ACL.

ADV FIELD



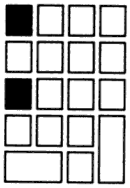
Completes the current ACE field and moves the cursor to the next ACE field, inserting text as needed.

BACKUP



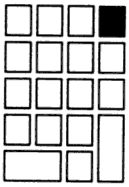
Sets the current direction backward for the FIND, FNDNXT, MOVE SCREEN, OVER ACE, and WORD keys. BACKUP means that movement will be toward the beginning of the ACL.

BOTTOM



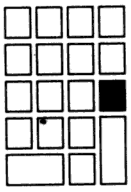
Sets the cursor position after the last line of the last ACE. Any entries you add will be placed at the end of the ACL.

DEL ACE



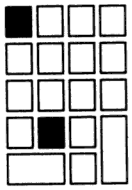
Deletes the entire ACE in which the cursor is positioned. The deleted ACE is placed in the delete-ACE buffer.

DEL C



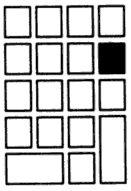
Deletes the character on which the cursor is positioned and saves it in the delete-character buffer.

DEL EOL



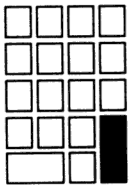
Deletes the remainder of the line from the current cursor position to the end of the line. The deleted text is stored in the deleted-line buffer.

DEL W



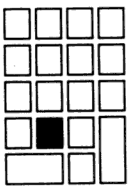
Deletes the text from the current cursor position to the beginning of the next word, and stores the text in the delete-word buffer.

ENTER



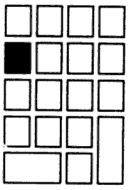
Indicates that the current ACE is complete and ready for parsing. This key terminates the insertion. You can press it while the cursor is located at any position within the ACE. (Pressing RETURN produces the same results.)

EOL



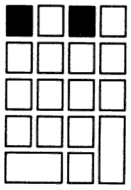
Moves the cursor to the end of the current line.

FIELD



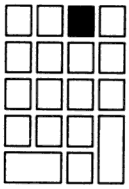
Completes the current ACE field and moves the cursor to the next ACE field or subfield, inserting text as needed.

FIND



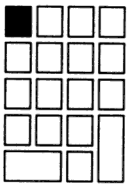
Searches for an occurrence of a string. Press the FIND key and then enter the string from the main keyboard. End the string by pressing either the ADVANCE or BACKUP key to set the direction of the search or the ENTER key to search in the current direction.

FNDNXT



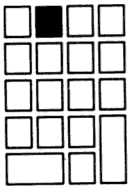
Searches for the next occurrence of the search string previously entered with the FIND key. The direction of the search is the current one.

GOLD



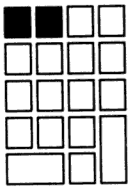
When pressed before another keypad key, specifies the second key's alternate function (the bottom function on the keypad diagram).

HELP



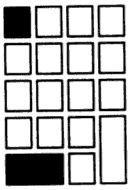
Use the HELP key to obtain help on the editing keypad.

HELP FMT



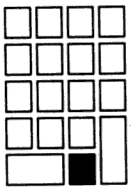
When you press the GOLD key followed by the HELP key, you can obtain help on ACE formats. The same result is produced by pressing the HELP key followed by the TAB key on the main keyboard.

INSERT



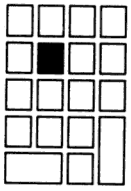
Indicates where an ACE is to be inserted. All the text from the current line is moved down one line, and a blank line is inserted.

ITEM



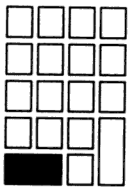
Selects the next item for the current ACE field.

MOVE SCREEN



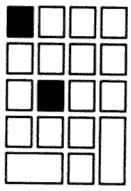
Moves the cursor one screen in the current direction (see ADVANCE or BACKUP). A screen is defined as two-thirds the number of lines in the display.

OVER ACE



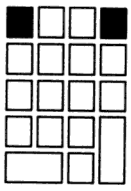
Moves the cursor to the beginning of the previous or next ACE, depending on the current direction.

TOP



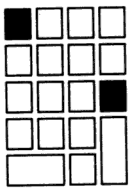
Moves the cursor position to the first character of the first ACE in the access control list.

UND ACE



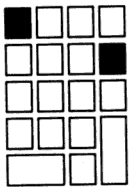
Inserts the contents of the delete-ACE buffer in front of the ACE in which the cursor is currently positioned.

UND C



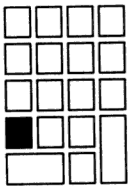
Inserts the contents of the delete-character buffer directly in front of the cursor.

UND W



Inserts the contents of the delete-word buffer directly in front of the cursor.

WORD



Moves the cursor forward or backward within the line by a word, depending on the current direction.

The following keys and key sequences may be used to move the cursor or shift the display window:

- [↓] —Moves the cursor to the character in the line below. If the access control entry in which the cursor is positioned is new, the ACL editor processes the access control entry before moving the cursor. If the entry is incomplete or improperly formatted, an error occurs and the cursor remains in place.
- [←] —Moves the cursor one character to the left. If the cursor is at the left margin, moves the cursor to the rightmost character in the line above.
- [→] —Moves the cursor one character to the right. If the cursor is at the right margin, moves it to the leftmost character in the line below.
- [↑] —Moves the cursor to the character in the line above it. If the access control entry in which the cursor is positioned is new, the ACL editor processes the access control entry before moving the cursor. If the entry is incomplete or improperly formatted, an error occurs and the cursor remains in place.
- [GOLD] [←] —Shifts the text in the display window 8 characters to the left.
- [GOLD] [→] —Shifts the text in the display window 8 characters to the right.

You can use the following keyboard keys to supplement the keypad keys. Keys in parentheses indicate the equivalent key for a VT200 series terminal.

- BACKSPACE (F12)—Moves the cursor to the beginning of the current line.
- DELETE (<X)—Deletes the character to the left of the cursor.
- LINE FEED (F13)—Deletes the text from the cursor to the beginning of the word. If the cursor is positioned at the first character of the word, deletes to the beginning of the previous word.
- TAB (TAB)—Moves the text to the right of the cursor to the next tab stop. Spaces are inserted to the right of the cursor to bring the cursor to the correct position.

3.2 VT200-Only ACL Editing Commands

On the VT200 series terminal, you can also use the following supplemental editing keypad keys:

- FIND (E1)—Elicits the *Search for:* prompt as the first step in the FIND operation. Type the search string after the prompt and then press either the DO or ENTER key to process the search. (Performs the same function as the FIND keypad key.)
- INSERT HERE (E2)—Indicates where an ACE is to be inserted, or, if support for the PASTE buffer is enabled, indicates the line where the selected text in the PASTE buffer is to be inserted. By default, support for the PASTE buffer is disabled; Section 3.2.1 explains how to enable the PASTE buffer.
- REMOVE (E3)—Removes the text within the select range to the PASTE buffer. Each time REMOVE is used, the previous contents of the PASTE buffer are deleted. By default, support for the PASTE buffer is disabled; Section 3.2.1 explains how to enable the PASTE buffer.
- COPY (GOLD E3)—Copies the text within the select range to the PASTE buffer. Each time COPY is used, the previous contents of the PASTE buffer are deleted. By default, support for the PASTE buffer is disabled; Section 3.2.1 explains how to enable the PASTE buffer.
- SELECT (E4)—Marks the beginning of a range of text to be removed or copied to the PASTE buffer. Press SELECT; move the cursor to include the desired amount of text to be removed or copied; and press either REMOVE (E3) or COPY (GOLD E3) to complete the operation. By default, support for the PASTE buffer is disabled; Section 3.2.1 explains how to enable the PASTE buffer.
- PREV SCREEN (E5)—Moves the cursor one screen in the backward direction. By default, a screen is defined as two-thirds the number of lines in the display.
- NEXT SCREEN (E6)—Moves the cursor one screen in the forward direction. By default, a screen is defined as two-thirds the number of lines in the display.

3.2.1 Using the PASTE Buffer

For VT200 series terminals, the ACL editor features a PASTE buffer that allows you to move sections of text from one part of the ACL to another. The commands used to move sections of the ACL in and out of the PASTE buffer include INSERT HERE, REMOVE, COPY, and SELECT and can be found on the supplemental editing keypad as keys E2, E3, GOLD E3, and E4, in that order.

By default, support for the PASTE buffer is disabled and can be turned on in one of the following manners:

- To enable the PASTE buffer for the current editing session, press CTRL/D. At the *TPU command:* prompt, type the following:

```
TPU command: ACLEDIT$X_PASTE_BUFFER:=1
```

- To permanently enable the PASTE buffer, you must change the value of the variable ACLEDIT$X_PASTE_BUFFER in the ACL editor section file from 0 to 1. You must then recompile the ACL section file; Section 6 describes how to modify and recompile this file.

If you enable the PASTE buffer, you must also switch the value of the variables ACLEDIT$X_CHECK_MODIFY from 1 (enabled) to 0 (disabled) using either of the methods previously listed. (This feature is not compatible with the use of the PASTE buffer.)

3.3 Control Key ACL Editing Commands

The following control keys also perform editing functions:

- CTRL/A —Determines whether characters are entered in insert mode or overstrike mode. Insert mode (the default) inserts a character to the left of the current character. Overstrike mode replaces the current character.
- CTRL/D —Allows the user to execute one TPU command.
- CTRL/H —Moves the cursor to the beginning of the line. (Performs the same function as the BACKSPACE key.)
- CTRL/J —Deletes the text from the cursor to the beginning of the word. (Performs the same function as the LINE FEED key.)
- CTRL/R —Refreshes the screen by deleting extraneous characters and restoring the previous display. (Performs the same function as the CTRL/W key.)
- GOLD CTRL/R —Returns the ACL to its original state before the ACL editor was invoked. (Performs the same function as GOLD CTRL/W.)
- CTRL/U —Deletes the text from the cursor to the beginning of the line.
- CTRL/W —Refreshes the screen by deleting extraneous characters and restoring the previous display. (Performs the same function as the CTRL/R key.)
- GOLD CTRL/W —Returns the ACL to its original state before the ACL editor was invoked. (Performs the same function as GOLD CTRL/R.)
- CTRL/Z —Ends the editing session and updates the ACL. (Unless otherwise specified, the recover and journal files [if used] are deleted.)

- GOLD CTRL/Z —Ends (quits) the editing session without saving any of the changes made to the object's ACL. (Unless otherwise specified, the recover and journal files [if used] are deleted.)

3.4 Terminating the ACL Editing Session

While you are editing, you can return to the original state of the ACL (before any changes were made), by entering GOLD CTRL/R. After you are satisfied with the results of the editing session, exit from the session by entering CTRL/Z. Any changes made to the object's ACL occur when the editing session ends. Because it is necessary to obtain exclusive access to a file to update an ACL, other users of a shared file may get a file-locked (or access conflict for directory files) error message when the ACL is updated at the end of the editing session.

If you do not want to incorporate the edits into the ACL, enter GOLD CTRL/Z. All edits made during the session are ignored, and the ACL returns to its state before the session.

4 Recovering an ACL Editing Session

By default, if an editing session ends abnormally, a journal file is created. The journal file name defaults to input-filename.JOU. You can use the /JOURNAL qualifier to specify an alternate journal file name. You can also use the /NOJOURNAL qualifier to override the default behavior and prevent the creation of a journal file.

To recover an editing session that has been interrupted, use the /RECOVER qualifier. If the journal file name is different than the default, you must specify it with /RECOVER.

For more information on any of these EDIT/ACL qualifiers, see the Qualifier Section.

5 Access Control Lists

This section describes the different types of ACEs that make up an access control list (ACL). The type of access protection needed determines the type of ACE used in a given situation. The three types of ACEs are:

- Identifier—Controls the type of access allowed to a particular user or group of users.
- Default protection—Defines the default protection for a directory, so that the protection can be propagated to the files and subdirectories created in that directory. (This type of ACE is applicable only to directory files.)
- Security alarm—Provides a security alarm when an object is accessed in a particular way.

The exact format of an ACE depends on its type, but all ACEs are enclosed in parentheses. In general, the format of an ACE is

```
(type[,options][,access_to_grant])
```

5.1 Identifier ACEs

An identifier ACE controls the types of access allowed to specific users based on the user's identification. The format for an identifier ACE is

```
(IDENTIFIER=identifier[,options][,access])
```

The first field in the identifier ACE consists of the keyword IDENTIFIER followed by one or more identifiers. An identifier can be one of the following types:

- User identification code (UIC)
- General, established by the system manager in the system rights database
- System-defined

A UIC can be in either numeric or named UIC format, as described in the *VAX/VMS DCL Concepts Manual.*

A general identifier, defined in the system rights database, is an alphanumeric string of 1 to 31 characters that must contain at least one alphabetic character. It can include the characters A through Z, dollar signs ($) and underscores (_), as well as the numbers 0 through 9.

The system manager creates and assigns the general identifiers and UICs to the system users with the Authorize Utility (AUTHORIZE).

System-defined identifiers are automatically defined by the system when the sytem manager creates a rights database. The following identifiers are system-defined identifiers:

BATCH	All attempts at access made by batch jobs
NETWORK	All attempts at access made over the DECnet–VAX network
INTERACTIVE	All attempts at access made by interactive processes
LOCAL	All attempts at access made by users logged in at local terminals
DIALUP	All attempts at access made by users logged in at dial-up terminals
REMOTE	All attempts at access made by users logged in via a network

In general, you should regard these six system-defined identifiers as mutually exclusive with each other and should not attempt to use them in combination with each other. However, you may combine them with other identifiers (UICs and general identifiers). When specifying multiple identifiers, connect them with plus signs (+).

The system takes the access action you include in the ACE only for the user who matches all the identifiers. For example, if you wanted to grant read access to user [301,25] running a batch job, you would specify the identifier ACE as

```
(IDENTIFIER=[301,25]+BATCH,ACCESS=READ)
```

Although it is not common practice for a number of users to share the same UIC, it is likely that a number of users will share the same general identifier. Users with the same general identifier do not need to be in the same UIC-based group. Furthermore, a single user can be associated with a number of different general identifiers as defined in the rights database. Because this is the case, the creator of an ACL has considerable flexibility in selecting sets of users and defining access capabilities for them.

For example, the user identified by the UIC [301,25] is a member of the UIC-based group 301. That user may be the only member of group 301 who is also associated with the general identifier PERSONNEL. An ACE defining a particular type of access for the users associated with the general identifier PERSONNEL, grants that access to that user, but not to the other members of group 301.

5.1.1 Specifying Options in Identifier ACEs

The options field in an identifier ACE controls whether an ACE is propagated, can be displayed, or can be deleted. This field in an identifier ACE begins with the keyword OPTIONS and takes one or more of the following keywords:

DEFAULT	Indicates that an ACE is to be included in the ACL of any files created within a directory. When the ACE is propagated, the DEFAULT indicator is removed from the ACL of the created file. This option is valid only for directory files. A default ACE does not grant or deny access; it just affects the ACL of new files.
HIDDEN	Indicates that this ACE should only be changed by the application that added it. The ACL editor will not permit modification or deletion. Thus, the ACL editor displays the ACE only to show its relative position within the ACL, not to facilitate editing of the ACE. The DCL DIRECTORY and SHOW ACL commands will not display hidden ACEs.
PROTECTED	Indicates that an ACE will be preserved even when an attempt is made to delete the entire ACL. A protected ACE must be specifically deleted with the ACL editor or by specifying the ACE on the command line of one of the ACL DCL commands.
NOPROPAGATE	Indicates that when copying an ACL from one version of a file to a later version of the same file, the ACE is not propagated.
NONE	Indicates that no options apply to an ACE. Although you may enter OPTIONS=NONE when you create the ACE, OPTIONS=NONE is not displayed when the ACE is displayed.

Multiple options are connected by plus signs (+). If you specify any other options with the NONE option, the other options will take precedence.

5.1.2 Specifying Access in Identifier ACEs

The third field in an identifier ACE specifies what type of access you are allowing the users identified in the first field of the ACE. This field begins with the keyword ACCESS followed by a string of access actions connected by plus signs. The following types of access are allowed in an identifier ACE:

READ	Accessor can read a file, read from a disk, or allocate a device.
WRITE	Accessor can read or write a file.
EXECUTE	Accessor can execute an image file or look up entries in a directory without using wildcard characters.
DELETE	Accessor can delete a file.
CONTROL	Accessor has all the privileges of the object's actual owner.
NONE	Accessor has no access to the object.

5.1.3 Sample Identifier ACEs

The most common type of ACL is one that defines the access to a file for a group of users. In the following sample ACL, access to a file is based on the identity of a user. PERSONNEL, SECURITY, and SECRETARIES are general identifiers that have been assigned to appropriate sets of users by the system manager using AUTHORIZE. NETWORK is a system-defined identifier, while [20,*] and [SALES,JONES] are examples of UIC identifiers.

```
(IDENTIFIER=SECURITY,OPTIONS=PROTECTED,ACCESS=READ+WRITE+EXECUTE+DELETE+CONTROL)
(IDENTIFIER=PERSONNEL,ACCESS=READ+WRITE+EXECUTE+DELETE)
(IDENTIFIER=SECRETARIES,ACCESS=READ+WRITE)
(IDENTIFIER=[20,*],ACCESS=READ)
(IDENTIFIER=NETWORK,ACCESS=NONE)
(IDENTIFIER=[SALES,JONES],ACCESS=NONE)
```

In the example above, the ACE that provides the greatest amount of file access is first in the list. In this way, if one of the users has both the SECURITY and PERSONNEL identifiers, the user obtains the maximum access rights through the first match, which is the SECURITY identifier. In this example, the user with UIC [SALES,JONES] will be prohibited from any access to the file, unless that user also happens to have one of the general identifiers (which is an oversight on the part of the creator of the ACL). If the ACL creator wants to be absolutely certain that the user with UIC [SALES,JONES] could not possibly gain access to the file, the ACE that is at the bottom of the ACL should be moved to the top.

Notice also that the order of the ACEs in the example above permits a number of users to gain types of file access over the DECnet–VAX network. The users with the identifiers of SECURITY, PERSONNEL, SECRETARIES, and UIC [20,*] can all gain some access over the network, although only those with the identifier SECURITY can gain full access. All other users are prohibited from network access due to the fifth ACE. While this may well be the intent of the ACL creator, it would be an unfortunate oversight if it were not. Always take special care in ordering the ACEs within your ACLs.

The only ACE that includes an option field is the first ACE, which contains the PROTECTED option. Use of this option guarantees that the first ACE will not be deleted, unless it is explicitly specified with the SET FILE/ACL /DELETE command or deleted using the ACL editor.

Identifier ACE for Directory

The OPTIONS=DEFAULT option of an identifier ACE allows users to define one or more default ACEs for inclusion in the ACLs for files created in a particular directory. A default ACE is supplied for all new files created in that directory; any existing files are not supplied with the default ACE. Thus, if you wanted all files in the directory [MALCOLM] to have an ACE that permitted read and write access to users with the PERSONNEL identifier, you could include the following ACE in the ACL for the file MALCOLM.DIR:

```
(IDENTIFIER=PERSONNEL,OPTIONS=DEFAULT,ACCESS=READ+WRITE)
```

As a result of this ACE, any file created in the [MALCOLM] directory has the following ACE:

```
(IDENTIFIER=PERSONNEL,ACCESS=READ+WRITE)
```

Notice that the DEFAULT option does not appear in the file's ACE. However, any subdirectory created in the MALCOLM directory has the DEFAULT option as part of its ACE so that the default ACE can be propagated throughout the entire directory tree.

Identifier ACE for Device

Identifier ACEs for devices are created in the same manner as other ACEs. For example, suppose your company has a special letter-quality printer (TTA8) that is used only for printing checks. As a result, the check forms are always loaded in the printer. This device is never to be used for logins and no queues are directed to it. Only one user, MGREY, is allowed read and write access to it. The system manager can accomplish this restriction by setting the protection on the printer with the command:

```
$ SET PROTECTION=(S,O,G,W)/DEVICE TTA8:
```

Then, the following identifier ACE, applied to the object TTA8, restricts access to the device:

```
(IDENTIFIER=MGREY,ACCESS=READ+WRITE)
```

5.2 Default Protection ACE

The default protection ACE is used to ensure that a particular type of UIC-based protection is propagated throughout a directory tree. This type of ACE allows you to specify protection for a particular directory structure that is different from the default protection applied to other directories. Default protection ACEs can be applied only to directory files.

The format for a default protection ACE is

```
(DEFAULT_PROTECTION[,options],protection_mask)
```

This type of ACE is specified by the keyword DEFAULT_PROTECTION. As in the identifier type of ACE, the second field in a default protection ACE, the options field, controls whether an ACE is propagated, can be displayed, or can be deleted. This field in a default protection ACE begins with the keyword OPTIONS and takes one or more of the following keywords:

HIDDEN	Indicates that this ACE should only be changed by the application that added it. The ACL editor will not permit modification or deletion. Thus, the ACL editor displays the ACE only to show its relative position within the ACL, not to facilitate editing of the ACE. The DCL DIRECTORY and SHOW ACL commands will not display hidden ACEs.
PROTECTED	Indicates that an ACE will be preserved even when an attempt is made to delete the entire ACL. A protected ACE must be specifically deleted with the ACL editor or by specifying the ACE on the command line of one of the ACL DCL commands.
NOPROPAGATE	Indicates that when copying an ACL from one version of a file to a later version of the same file, the ACE is not propagated.
NONE	Indicates that no options apply to an ACE. Although you may enter OPTIONS=NONE when you create the ACE, OPTIONS=NONE is not displayed when the ACE is displayed.

Multiple options are connected by plus signs (+). If you specify any other options with the NONE option, the other options will take precedence.

The protection mask is specified the same as for UIC-based protection with the user categories—SYSTEM, WORLD, GROUP, and OWNER—and the access categories—READ, WRITE, EXECUTE, and DELETE. See the discussion of UIC-based protection in the *VAX/VMS DCL Dictionary* for more information.

The following ACE, included in an ACL for the directory MALCOLM, sets up default protection so that any files created in the directory will allow the system and owner groups read, write, execute, and delete access and the group and world groups no access:

```
(DEFAULT_PROTECTION,S:RWED,O:RWED)
```

Note that when you add or change the default protection for a directory, there is no effect on the files already created in the directory; however, all new files created in the future will receive the default protection.

You might also choose to have the default protection ACE PROTECTED, so that its ACE is preserved if an attempt is made to delete the entire ACL. To accomplish this, you could create the following ACL:

```
(DEFAULT_PROTECTION,OPTIONS=PROTECTED,S:RWEDC,O:RWEDC,G,W)
```

5.3 Security Alarm ACE

The security alarm ACE allows you to specify that an alarm message be sent to the security operator's terminal if a certain type of access takes place.

Whenever it is important that some response be taken when an object has been accessed in a particular unauthorized way, you should consider adding a security alarm ACE to the object's ACL. The security alarm ACE specifies the type of access to the object that you regard significant enough to warrant an alarm message being sent to the operators enabled as security operators.

The security action depends on whether the alarms to operators enabled as security operators have been enabled through the DCL command SET AUDIT.

Although you can independently create alarm ACEs in an ACL, causing the system to observe the event and take the required action, you should depend on some additional coordination with your system's security manager (the person who possesses the SECURITY privilege). The security manager is responsible for enabling the alarm feature. Since this feature uses system resources, the security manager may be reluctant to leave it enabled at all times.

The format of a security alarm ACE is

```
(ALARM_JOURNAL=SECURITY[,options][,access])
```

This type of ACE is specified by the keywords ALARM_JOURNAL=SECURITY. As in the IDENTIFIER type of ACE, the second field in security alarm ACE begins with the keyword OPTIONS, which takes one or more of the following keywords:

Description

DEFAULT	Indicates that an ACE is to be included in the ACL of any files created within a directory. When the ACE is propagated, the DEFAULT indicator is removed from the ACL of the created file. This option is valid only for directory files.
HIDDEN	Indicates that this ACE can only be changed by the application that added it. The ACL editor will not permit modification or deletion. Thus, the ACL editor displays the ACE only to show its relative position within the ACL, not to facilitate editing of the ACE. The DCL DIRECTORY and SHOW ACL commands will not display hidden ACEs.
PROTECTED	Indicates that this ACE will be preserved even when an attempt is made to delete the entire ACL. A protected ACE must be explicitly deleted with the ACL editor or by specifying the ACE on the command line of one of the ACL DCL commands.
NOPROPAGATE	Indicates that when copying an ACL from one version of a file to a later version of the same file, the ACE is not propagated.
NONE	Indicates that no options apply to this ACE. (Although you enter OPTIONS=NONE when you create the ACE, OPTIONS=NONE is not displayed when the ACE is displayed.)

Multiple options are connected by plus signs (+). If you specify any other options when specifying NONE, the other options take precedence.

The third field in an alarm ACE controls the type of access that causes the alarm to be sent. You can specify any of the following access actions with the ACCESS keyword:

READ	Generates an alarm if an accessor attempts to read the object
WRITE	Generates an alarm if an accessor attempts to read or write the object
EXECUTE	Generates an alarm if an accessor attempts to execute the object
DELETE	Generates an alarm if an accessor attempts to delete the object
CONTROL	Generates an alarm if an accessor attempts to perform control operations on the object, such as changing the protection on the object
SUCCESS	Generates an alarm for each successful attempt by an accessor to access the object with the access-allowed-mask entries from the set above
FAILURE	Generates an alarm for each unsuccessful attempt by an accessor to access the object with the access-allowed-mask entries from the set above

For an alarm to have any effect, you must include either SUCCESS or FAILURE or both.

6 Customizing the ACL Editor

The current version of the ACL editor was written using the VAX Text Processing Utility (VAXTPU). This provides an environment that is user extensible; that is, users can modify the ACL editor to meet their own needs. Users may modify and recompile the ACL section file SYS$LIBRARY:ACLEDIT.TPU (the source file that creates the compiled ACL section file SYS$LIBRARY:ACLSECINI.TPU$SECTION) or create their own ACL section file.

You should be familiar with the VAXTPU language before attempting to customize the default ACL section file or design a new ACL section file. Chapter 5 of the *VAX Text Processing Utility Reference Manual* contains information on writing and processing a section file.

6.1 Modifying Variables in the ACL Section File

Following is a list of the variables available through the ACL section file and their defaults.

Variable	Meaning
ACLEDIT$X_CHECK_DUPLICATES	Controls whether or not a check for duplicate ACEs is made. This variable may take the following values: 0 No duplicate ACE check is made. 1 A duplicate ACE check is made. If the ACE to be entered matches an existing ACE, an error message is returned. This is the default.
ACLEDIT$X_CHECK_MODIFY	Allows or disallows modification of ACEs. This variable may take the following values: 0 The ACE may be modified. 1 The ACE may not be modified. If an attempt is made to modify the ACE, it is replaced with the original ACE. This is the default.
ACLEDIT$X_DIRECTORY_FILE	Indicates whether or not the object is a directory file. This variable may take the following values: 0 The object is not a directory file. 1 The object is a directory file.
ACLEDIT$X_PASTE_BUFFER	Controls whether or not PASTE buffer support is enabled for VT200 series terminals. This variable may take the following values: 0 PASTE buffer support is disabled. This is the default. 1 PASTE buffer support is enabled.

Variable	Meaning
ACLEDIT$X_PROMPT	Controls whether or not automatic text insertion (prompt mode) is enabled. This variable may take the following values: 0 Prompt mode is disabled. 1 Prompt mode is enabled. This is the default.
ACLEDIT$X_USE_DEFAULT_OPT	Controls whether or not the DEFAULT option may be used with non-directory ACEs. This variable may take the following values: 0 The DEFAULT option can only be used with ACEs of directory (.DIR) files. This is the default. 1 The DEFAULT option is available for use with ACEs of all object types.
ACLEDIT$C_WINDOW_SHIFT	Specifies the number of columns to shift the edit window in the desired direction, GOLD ← for a left shift and GOLD → for a right shift. The default is 8 columns.

If you modify any of the variables listed above or change any other part of the ACL section file, you must recompile the section file with the command:

```
$ EDIT/TPU/NOSECTION/COMMAND=SYS$LIBRARY:ACLEDIT
```

You use the previous command if you make changes directly to the source code file (SYS$LIBRARY:ACLEDIT) that creates the compiled ACL section file SYS$LIBRARY:ACLSECINI. If you are adding a user-written command file to the existing ACL section file, recompile the section file with the command:

```
$ EDIT/TPU/SECTION=SYS$LIBRARY:ACLSECINI/COMMAND=your_section.TPU
```

The resulting compiled VAXTPU ACL section file will be placed in your current directory. To use the new section file, you must do one of the following:

- Move the resulting compiled section file, ACLSECINI.TPU$SECTION, to the SYS$LIBRARY directory. This will change the default ACL editor section file for all users.
- Define the logical name ACLSECINI in your LOGIN.COM file to point to the resulting compiled section file with the following command:

  ```
  $ DEFINE ACLSECINI yourdisk:[yourdir]ACLSECINI
  ```

Note that the default file type for section files before compiling (the source file) is TPU and the default file type for the compiled section file is TPU$SECTION. See Chapter 5 of the *VAX Text Processing Utility Reference Manual* for more information on writing and processing a VAXTPU section file.

6.2 Using the ACL Editor CALL_USER Routine

The ACL editor CALL_USER routine is part of the shareable image SYS$LIBRARY:ACLEDTSHR.EXE. You can incorporate the ACL editor CALL_USER routine with its existing function codes into your own ACL section file, or you can write your own CALL_USER routine that recognizes a different set of function codes.

The ACL editor CALL_USER routine recognizes only those functions used by the ACL editor VAXTPU section file. All other function codes are passed to a user-supplied CALL_USER routine; if the high-order word of the CALL_USER function code (bits <16:31>) contains the ACL editor facility code, it is handled by the ACL editor CALL_USER routine. Otherwise, an attempt is made to locate a user-supplied CALL_USER routine. Refer to the description of the CALL_USER routine in the *VAX Text Processing Utility Reference Manual* for more information on creating your own CALL_USER routine.

Following is a description of the CALL_USER routine function codes supported by the ACL editor:

Function Code	Mnemonic	Description
18153473	ACLEDIT$C_PARSE_ACE	Parses the input string (ACE) and returns the parsed (binary) ACE if no errors are found. Otherwise, the returned string contains a zero as the first two characters and the unparsed portion of the input ACE as the remainder of the string.
18153474	ACLEDIT$C_CHECK_MODIFY	Returns the string "READ_WRITE" if the ACE may be modified by the user. Otherwise, returns the string "READ_ONLY."
18153475	ACLEDIT$C_PROMPT_MODE	Returns the string "PROMPT_MODE" if the prompt mode option was specified. Otherwise, returns the string "NOPROMPT_MODE."
18153476	ACLEDIT$C_CHECK_ACE	Parses the input string (ACE) and returns the parsed (binary) ACE if no errors are found. Otherwise, the ACE text is highlighted in reverse video and a VAXTPU variable of the form ACLEDIT$X_RANGE_x is created to identify the ACE in error. (The "x" is a sequential number starting with 1.)
18153477	ACLEDIT$C_CHECK_DIR	Returns the string "DIRECTORY_FILE" if the object being edited is a directory file. Otherwise, returns the string "NODIRECTORY_FILE."
18153478	ACLEDIT$C_SET_CANDIDATE	Parses the input string (ACE) and returns the string "PARSE_OK" if no error was encountered. Otherwise, returns the string "PARSE_ERROR." If the parse was successful, a check is then made for duplicate ACEs using the CALL_USER function ACLEDIT$C_CHECK_DUP.
18153479	ACLEDIT$C_CHECK_DUP	Parses the input string (ACE) and returns the string "PARSE_ERROR" if an error was encountered. Otherwise, the parsed (binary) ACE is compared with the candidate ACE set by the CALL_USER function ACLEDIT$C_SET_CANDIDATE. Returns the string "DUPLICATE_ACE" if the ACE is a duplicate or "UNIQUE_ACE" if it is not a duplicate.

COMMAND QUALIFIERS

The EDIT/ACL command qualifiers provide you with control over journaling and recovering the editing session.

/JOURNAL

Controls whether a journal file is created for the editing session.

FORMAT

/JOURNAL *[=file-spec]*
/NOJOURNAL

DESCRIPTION

By default, the ACL editor keeps a journal file that contains a copy of the ACL editor's keypad actions with each modification made during the editing session. The /JOURNAL qualifier controls the creation of a journal file for the editing session. If you specify /NOJOURNAL, no journal file is generated. If you omit the qualifier or specify /JOURNAL, a journal file is created. If you omit the journal file specification, by default the journal file is named input-file-spec.JOU.

No wildcard characters are allowed in the journal file specification.

If your editing session ends abnormally, you can invoke the ACL editor again and recover the changes made during the aborted session by specifying the /RECOVER qualifier and providing the name of the journal file if it differs from the default name.

EXAMPLE

```
$ EDIT/ACL/JOURNAL=COMMONACL.SAV MECH1117.DAT
```

With this command, the user creates a journal file named COMMONACL.SAV that contains a copy of the ACL and the editing commands used when creating the ACL for the file MECH1117.DAT. The file COMMONACL.SAV can then be used for recovery of the editing session for MECH1117.DAT's ACL, should this session terminate unexpectedly. A journal file is created by default, but the qualifier is specified because COMMONACL.SAV has a more descriptive name than the default journal file of MECH1117.JOU.

/MODE

Specifies if prompting is to be used during the editing session.

FORMAT

/MODE=*option*

DESCRIPTION

By default, the ACL editor selects prompt mode. Use the /MODE qualifier to specify one of these prompt options:

PROMPT	Specifies that where possible the selected field within the ACE is initially filled with the first of a list of items that may apply to the field.
NOPROMPT	Specifies that no prompting is used by the ACL editor.

EXAMPLE

```
$ EDIT/ACL/MODE=NOPROMPT WEATHERTBL.DAT
```

With this command, the user initiates an ACL editing session to create an ACL for the file WEATHERTBL.DAT. The /MODE=NOPROMPT qualifier specifies that no assistance is required in entering the ACL entries.

/OBJECT

Specifies the type of the object whose ACL is being edited.

FORMAT

/OBJECT=*type*

DESCRIPTION

By default, the ACL editor assumes that the object whose ACL is being edited is a file. If the object is not a file, the /OBJECT qualifier is required. The following keywords may be specified with /OBJECT:

FILE	Specifies that the object type is a file or a directory file.
DEVICE	Specifies that the object type is a device.
SYSTEM_GLOBAL_SECTION	Specifies that the object type is a system global section.
GROUP_GLOBAL_SECTION	Specifies that the object type is a group global section.
LOGICAL_NAME_TABLE	Specifies that the object type is a logical name table.

EXAMPLE

```
$ EDIT/ACL/OBJECT=DEVICE WORK1
```

The /OBJECT qualifier is used in this command because the object whose ACL is being edited is a device.

/RECOVER

Determines whether or not the ACL editor restores the object's ACL from a journal file prior to starting the editing session.

FORMAT

/RECOVER *[=file-spec]*
/NORECOVER

DESCRIPTION

The /RECOVER qualifier specifies that the ACL editor should restore the ACL from the journal file specified by input-file-spec.JOU. This operation restores the ACL to the state it was in when a previous ACL editing session ended abnormally.

If the journal file has a file name other than input-file-spec.JOU, specify it with the /RECOVER qualifier.

EXAMPLE

```
$ EDIT/ACL/JOURNAL=SAVEACL MYFILE.DAT
    .
    .
    .
User creates ACL until system crashes
    .
    .
    .
$ EDIT/ACL/JOURNAL=SAVEACL/RECOVER=SAVEACL MYFILE.DAT
    .
    .
    .
ACL is restored and user proceeds with editing until done
    .
    .
    .
  ^Z
$
```

The user initiates the ACL editing session by specifying that the journal file SAVEACL.JOU be saved if the session ends abnormally. The session proceeds until aborted by a system crash. To recover from the aborted editing session, and continue with additional edits, the user issues the second EDIT/ACL command. This session begins by restoring the session with the journal SAVEACL.JOU. Once the editing session proceeds to a normal completion (with a CTRL/Z), the journal file SAVEACL.JOU is deleted.

EXAMPLE

```
$ EDIT/ACL MYFILE.DAT
(IDENTIFIER=[360,7],ACCESS=NONE)
(IDENTIFIER=[360,*],ACCESS=READ+WRITE)
    .
    .
    .
User edits the ACL.
    .
    .
    .
   ^Z
$
```

In this example the owner of the file MYFILE.DAT wants to modify the file's access control list. A journal file called MYFILE.JOU is automatically created to record the ACL. By default the editor enters prompt mode. The system displays the two existing ACEs in this ACL.

The user wants to change the UIC from [360,7] to [360,17]. The user makes the correction, presses ENTER on the keypad, and then exits with CTRL/Z. Since this is a normal exit, the journal file MYFILE.JOU is automatically deleted.

Index

A

E

H

I

J

K

M

O

R

S

VAX/VMS
Access Control List
Editor Reference Manual
AA–Z414B–TE

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL
FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line

VAX/VMS
Access Control List
Editor Reference Manual
AA–Z414B–TE

READER'S COMMENTS

Note: This form is for document comments only. DIGITAL will use comments submitted on this form at the company's discretion. If you require a written reply and are eligible to receive one under Software Performance Report (SPR) service, submit your comments on an SPR form.

Did you find this manual understandable, usable, and well organized? Please make suggestions for improvement.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Did you find errors in this manual? If so, specify the error and the page number.

_____

_____

_____

_____

_____

_____

_____

_____

_____

Please indicate the type of user/reader that you most nearly represent:

- ☐ Assembly language programmer
- ☐ Higher-level language programmer
- ☐ Occasional programmer (experienced)
- ☐ User with little programming experience
- ☐ Student programmer
- ☐ Other (please specify) _____

Name _____ Date _____

Organization _____

Street _____

City _____ State _____ Zip Code _____
or Country

Do Not Tear - Fold Here and Tape

digital

No Postage Necessary if Mailed in the United States

BUSINESS REPLY MAIL

FIRST CLASS PERMIT NO.33 MAYNARD MASS.

POSTAGE WILL BE PAID BY ADDRESSEE

SSG PUBLICATIONS ZK1-3/J35
DIGITAL EQUIPMENT CORPORATION
110 SPIT BROOK ROAD
NASHUA, NEW HAMPSHIRE 03062-2698

Do Not Tear - Fold Here

Cut Along Dotted Line

NOTES

NOTES